あふれのない浮動小数点表示について (数値計算のアルゴリズムの研究)

AUTHOR(S):

伊理, 正夫; 松井, 正一

CITATION:

伊理, 正夫 ...[et al]. あふれのない浮動小数点表示について (数値計算のアルゴリズムの研究). 数理解析研究所講究録 1980, 382: 112-134

ISSUE DATE:

1980-04

URL:

http://hdl.handle.net/2433/104830

数理解析研究所講究録
382巻 1980年 112-134

あふれのない浮動小数点表示について

東京大学 工学部 伊理 正夫
松井 正一

要旨

従来の浮動小数点表現方式で問題となっていた指数部あふれを解決する新しい表現方式の提案を行う．新表現方式は，指数部と仮数部との境界を動的に変化させることにより，指数部あふれを防ぐとともに，普通の大きさの数に対しこれまでより高い精度を確保することができる．また普通の数ではない数をいくつか考えることにより，"数の体系"を閉じたものとする．新しい表現方式が効果を発揮するような計算例も示す．

1. はじめに

数値計算に計算機を使う際には，実数(の近似値)を一語中に"浮動小数点"の形で表すのが慣習になっている．初期の計算機では"固定小数点"が用いられたが，"桁取り(スケーリング)"の繁雑さのために，有効桁数の少々(指数部に割当てる分)の犠牲を払っても任意の大きさの数が同じ相対精度で取扱える浮動小数点方式に完全に軍配が上がった．浮動小数点方式にも多くの変種があるが，最近では，IBM型の16進，仮数部絶対値，指数部クビット"げたばき"式の型が優勢である．経験が積まれるに従って，このIBM型に対する多くの不満が数値計算家の間に充ちている．

最も切実な問題の一つに"あふれ(overflow, underflow)"がある．一松の解説[6]にもこのような問題の指摘がある．一般に，Hamming[4]も述べているように，計算が進むにつれて扱われる数の指数部の絶対値の範囲は広がるので，扱える数の範囲がたとえば$(16^{-26}, 16^{26}) \fallingdotseq (10^{-77}, 10^{77})$のように，予め限られていると，それからはみ出す数が生じ易い．それを防ぐために計算途上で数の大きさを常に検査するのでは「スケーリングの心配無用」という浮動小数点方式の最大の特長が失なわれてしまう．一松[6]に紹介されているKahanの提案は一言でいえば，(i)一語中で指数部に割当てる部分を増大させ，(ii)"非数"を導入してあふれた後の面倒をもみることにある．

その他，記数法の底(=2,4,8,16等)の良し悪しについての議論[1],[2](およびそれらの参考文献)，通常のものとは異なる表示体系の提案[3],[12]等々，浮動小数点方式の本質的な問題への関心が最近高まりつつあるようにみえる．

可変語長にすれば問題はすべて解決するという議論もあるが[9]，非常に特殊な目的を除けば，通常の数値計算では一つの数値の担うべき"情報量"には当然限度があり，計算時間，記憶場所の経済性の点からも，これらは考察の外としてよいであろう．

本論文では，「固定語長内で実数を近似的に表現する」という大前提のもとで，一つの理想的な方式を提案し，その特徴を調べ，それを具体的に実現し，それによる数値計算例を示す．

本論文で提案する方式の基本的な考え方を形式ばらずに述べると次の2点となる．(i)与えられた一語中に数を表現する際，より重要な情報から順に必要なビットを割当てる(指数部に必要なだけのビット数をまず確保した後に，残りを仮数部に用い，指数部だけでも入り切らなくなったら，指数部を更に浮動小数点表示にして，"指数部の指数部"に優先的な割当てをする，以下同様なことを繰り返す)．(ii)数値演算の結果が常にプログラムで取扱い可能な形である(あふれによる割込みが生じたりしない)ために導入すべき必要にして十分な"非数"をも含んだ"数の体系"を確定させる．(この際四則演算に関する対称性を重視する．)

(i)については Morris [10] に類似の着想が認められ，また Knuth [9] の演習問題（第4章 2.1 節の 17 番）にそのような方向への示唆があるが，本論文ではその可能性を積極的に追求してみる．(ii) は (i) とほとんど独立な考え方で，Kahan 等が部分的に論じているが [8]，いくつかの妥当と考えられる前提のもとで必然的に一つの体系が確定されることを指摘する．

2. 基本的な考え方

2.1. 指数部可変長方式とその一般化

実数 x の浮動小数点表現は，底を β ($=2, 4, 10, 16$ 等) とするとき，

$$\text{<レベル0>} \qquad x = (-1)^{s_0} F \times \beta^{(-1)^{s_1} E_1} \qquad (2.1)$$

の形をとる．s_0, s_1 は 0 あるいは 1，E_1 は非負整数で，F は正規化条件

$$1 > F \geqq 1/\beta \qquad (2.2)$$

あるいは

$$\beta > F \geqq 1 \qquad (2.3)$$

を満たす実数である．(通常は前者が多く用いられているが，ここでは §2.3 で述べる要求をも考えて後者を採用する．) $x \neq 0$ に対して s_0, E_1, F は ($E_1 \neq 0$ なら s_1 も) 一意に定まる．固定語長の一語中に E_1 と F を表現する際いくらかの近似誤差が不可避であるとすれば，その誤差は F のみに負わせるのが明らかによい．(正規化

条件の範囲でFが変化してもE_1が±1変化するよりも，もたらす影響は小さいから．）

$|\log_\beta|x||$が小さいうちはE_1の桁数は少いから，E_1は少いビット数で正確に表せ，一語中の大きな部分をFに割当てることができる．$|\log_\beta|x||$が大きくなるにつれて，E_1の桁数が増すので，それに割当てる部分が増加し，Fに割当てる部分は次第に減少する．

E_1だけで一語を占有するくらいに$|\log_\beta|x||$が大きくなった後は，(2.1)の代わりに，

$$\langle \text{レベル}1 \rangle \quad x=(-1)^{\Delta_0} F \times \beta^{(-1)^{\Delta_1} E_1} \times \beta^{E_2} \tag{2.4}$$

という表示を採用する．このときには，Fは"正規化条件を満たすある数"という情報しか有しない（これを表すための場所は不用）．E_1, E_2は(2.1)の表示をしたときのE_1（これをEと書く）が

$$E_1 \times \beta^{E_2} \leq E < E_1 \times (\beta^{E_2+1} - 1) \tag{2.5}$$

を満足し，かつE_1, E_2が一語中に収まる範囲内でE_1の桁数がなるべくなるべく大きくなるように選ぶ．

同様にして，更に高次の"レベル"の表示に進むことができる．すなわち(2.4)においてE_2が一語を占めるに至った後には，

$$\langle \text{レベル}2 \rangle \quad x=(-1)^{\Delta_0} F \times \beta^{(-1)^{\Delta_1} E_1} \times \beta^{E_2 \times \beta^{E_3}} \tag{2.6}$$

という表示を採用する。ここで，Fは"正規化条件を満たすある数"，E_1は"条件

$$1 \leq E_1 < \beta^{E_2} \times \beta^{E_3} - 1 \qquad (2.7)$$

を満たすある数"，E_2，E_3は(2.1)の表示をしたときのE_1(これをE_2と記す)が

$$E_1 \times \beta^{E_2} \times \beta^{E_3} \leq E < E_1 \times \beta^{E_2} \times (\beta^{E_3} - 1) \qquad (2.8)$$

を満足し，かつE_2とE_3が一語中に収まる範囲内でE_2の桁数がなるべく大きくなるように選ぶ．レベル3，4…も同様に定める．

実際には，

(i) どのレベルで表示しているか，

(ii) FとEに割当てられている場所の境界(レベル0の場合，E_iとE_{i+1}の境界(レベル i(≥ 1)の場合)はどこか，

等のための情報も同じ語中に収めねばならない．

2.2. 底の選択

浮動小数点表示の底βについては，いろいろな観点から現在ではβ=2(2進法)を採用するのが最良であるとされている[1][2]．また，「正規化数の先頭ビットが1」ということを利用してビット数の複約を企れることも見逃せない長所である．

2.3. 符号付き絶対値表示の採用

上記の§2.1，§2.2の原則の長所を発揮させるためには，(2.1)の仮数部±F，指数部±Eは，"補数表示"，"ゲたばき表示"等ではなく，

符号付き絶対値表示でなければならない。これを正規化条件(2.3)と組み合わせることにより，数の表示法にとって望ましい[11]，加法逆元，乗法逆元に関する対称性も確保できる。(指数部の絶対値が最大の数ではF=1であることに注意。)

2.4. 丸めの方式

丸めの方式は特殊な目的(たとえば区間演算)を除けば四捨五入(2進なら0捨1入)が良いから，これを採用する。他の丸めの方式でも本質的な差異は生じない。以下では"0捨1入"を前提とする。

2.5. "非数"の役割

固定長の一語で区別できる数の個数は有限であるが，一方ではいくらでも大きい(小さい)数が現れうるので，"ある数よりも大きい(小さい)数"というようなものも一つの"数"として扱わざるを得ない。これはいわば"±∞"のようなものである。これが不可避であるならば，これを積極的に"数"の体系に組み入れて，四則演算に関して閉じた，"あふれ"が起きてもOSの厄介にならない体系を考えるのがよい。(このような方向への試みはKahan[8]にも見られるが，いくらかの曖昧性が残されている。)四則演算に関する対称性を要請すれば，"非数"の体系は一意に確定する(§4を参照)。

3. 新浮動小数点方式の実現法

レベルを無限にとることは不可能であり，実際的にはどこかで打切らね

ばならない．本節ではレベル0の実現についてまず考察する．より高次のレベルについてもほぼ同じ方針で実現可能であろう．

3.1. レベル0の実現法

2進法で正規化条件(2.3)のもとで，(2.1)の形に数x(≠0)を表わせば，次の形のビット並びになる：

$$F = 1.f_2 f_3 \cdots f_m \qquad (3.1)$$

$$E = 1e_{n-1}e_{n-2}\cdots e_1 \quad (\text{あるいは } E=0) \qquad (3.2)$$

そこで数xを表わす情報としては，次の $L = m+n$ ビットが必要である：

$$s_0 f_2 f_3 \cdots f_m e_{n-1} e_{n-2} \cdots e_1$$

($n=1$ のとき $E=1$; $n=0$ のときは $E=0$ で s_1 は不用)．Lが固定されていれば，nは 0, 1, …, L-1 のいずれかの値をとる．仮数部Fと指数部Eとの"境界"を示すこの値nのために，$\lceil \log_2 L \rceil$ ビットがさらに必要となる．そこで，

$$L + \lceil \log_2 L \rceil \qquad (3.3)$$

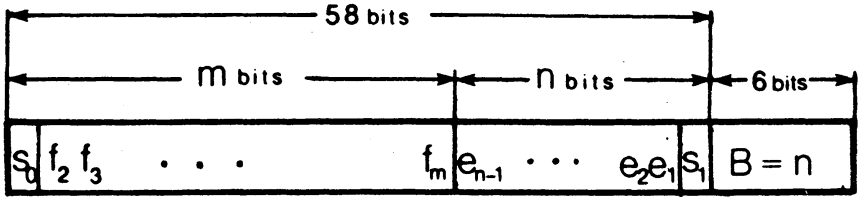
が一語長を超えないようにLを選べばよい．数xに対して，このような表現が一意に定まることは明らかである．(なお，理想的な"0"は本体系では"非数"として扱う．§4を参照．)

3.2. 具体的な実現例

一語64ビットの場合を具体的に考える．(3.3)の条件から，$L=58$, $\lceil \log_2 L \rceil = 6$ として，図-1のようなデータ構造を得る．

仮数部(の絶対値)の長さは58−1ビット(明示しない先頭ビット

も含む)，指数部(符号も含む)の長さは0～57ビット(B=nキ0のとき明示していない先頭ビットも含む)となり，|x|が約$10^{-10^{16}}$から$10^{10^{16}}$までの数xがレベル0で表現可能である．(表現誤差については§5参照.) Bの値のうち58～63は無意味となるが，それらは(i)非数，(ii)高次レベルの数，等を表わすのに使用できる．



仮数部$(-1)^{s_0}F=(-1)^{s_0}\times(1.f_2f_3\cdots f_m)_2$，指数部$(-1)^{s_1}E=(-1)^{s_1}\times(1e_{n-1}\cdots e_2e_1)_2$

B=nは仮数部と指数部の境界を表わす．(B=0, 1, …, 57; B=0のときE=0)

図-1. 新しい表現の浮動小数点数のデータ構造(レベル0)

このような(内部)表現を持つ(レベル0)の出力に際しては，印字巾のみを指定し，その内で指数部(10進整数)の場所をまず確保し，残りを(X.XX…　の形に正規化された)仮数部に割当てればよい．(具体的には§6の例を参照.)

数の入力の場合の表現は，従来許されている諸形式をすべて許容することとする．

4. 非数を含んだ数の体系

計算機で数値計算を行う際に扱う"数"は少くとも四則演算に

関して閉じた体系をなしている（演算が"合法的でない"ような被演算数の組があってもプログラムの正常な流れがそれによって妨げられない）ことが望ましい．そもそも"一語中に表現された数"は，数直線上の一点というよりは，ある区間（区間内数におけるようにそれを明示するかどうかを別として）を表わすと考えるべきである．§2.3で触れた四則演算に関する対称性の観点からすると，我々がまず考えるべき実数は $R^+ \cup R^-$ である．ここで

$$R^+ = \{\text{正の実数}\},\quad R^- = \{\text{負の実数}\}. \qquad (4.1)$$

§3の表現法では $R^+ \cup R^-$ の一部分が表現可能である．考えるべき区間の"型"としては少くとも次のもの（および $\leq$ をくで置きかえたもの）が必要である．

(i) $\{x \mid a \leq x \leq b\ ;\ a,b \in R^+\}$, $\{x \mid a \leq x \leq b\ ;\ a,b \in R^-\}$;

(ii) $\{x \mid a \leq x\ ;\ a \in R^+\}$, $\{x \mid x \leq a\ ;\ a \in R^-\}$;

(iii) $\{x \mid x \leq a\ ;\ a \in R^+\}$, $\{x \mid a \leq x\ ;\ a \in R^-\}$;

(iv) $\{x \mid x \leq a \text{ or } b \leq x\ ;\ a \in R^-,\ b \in R^+\}$;

(v) $\{x \mid a \leq x \leq b\ ;\ a \in R^-,\ b \in R^+\}$;

(vi) R^+, R^-;

(vii) $R\ (= R^+ \cup R^- \cup \{0\})$.

なぜならば，基本的な型(i)から出発して演算を繰返してゆくと上記のすべての型が生起する可能性があるからである．これらを，それぞれ

(i) $+num$, $-num$;

(ii) $+\infty$, $-\infty$;

(iii) $+0$, -0;

(iv) ∞;

(v) 0;

(vi) $+?$, $-?$;

(vii) $?$

と略記し(図-2), 型(i)以外のものを"非数"と呼ぶ.

§3の表現方式で表現可能な数±numの代表点であるとみなし, それらの間の演算は, 代表点間の普通の意味での演算と考え, 演算結果は,

a. §3の方法で表現可能ならば(必要なら丸めを行なった後)±num,

b. 絶対値が大きすぎて表現不能ならば$\pm\infty$,

c. 絶対値が小さすぎて表現不能ならば,

c1. 仮数部に符号の情報が残っていれば± 0,

c2. 仮数部に符号の情報が残っていなければ0,

であると約束する. その他の非数がどのようにして出現するかを, また型(i)~(vii)を考えれば四則演算に関して閉じた体系が(厳密には"最小の体系")が得られることを, 表-1~表-4に示す.

数(および非数)xとyの大小関係は, 上記a, b, cおよび表-2の減算の結果の約束を用いて, $x-y$の演算結果が$+$(あるいは$-$)の符号を有するとき$x>y$($x<y$)であると約束する. $x=y$はxとyが

同一のビットパターンで表現されているときのみに限ることとする．こうすれば，たとえば「$-\infty < -num < -0 < +0 < +num < +\infty$」，「$-num < 0 < +num$」，「$x < y$，ならば $x+u < y+v$」等は成立する．しかし，「$x = y$，ならば $x+u < y+v$」等が成立するとは限らない．

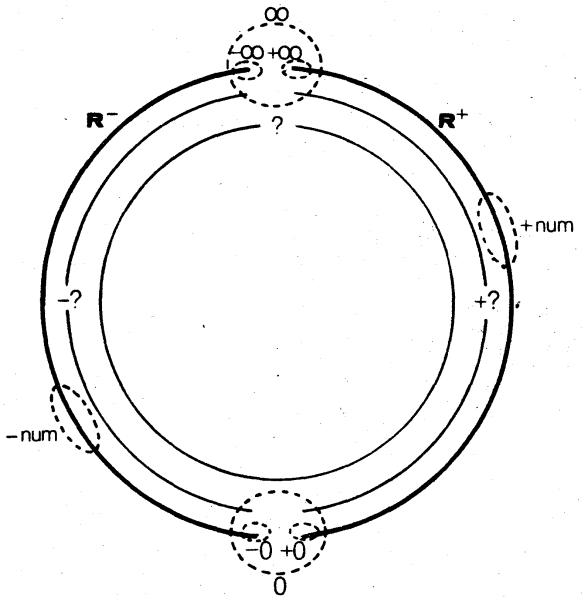


図-2. 四則演算に関して閉じた数の体系の概念図

表-1. 加算の定義

op1 \ op2	+num	-num	+∞	-∞	+0	-0	+?	-?	∞	?	0
+num	+num +∞	±num ±0,0	+∞	-∞	+num	+num	+?	?	∞	?	+num
-num	±num ±0,0	-num -∞	+∞	-∞	-num	-num	?	-?	∞	?	-num
+∞	+∞	+∞	+∞	?	+∞	+∞	+?	?	?	?	+∞
-∞	-∞	-∞	?	-∞	-∞	-∞	?	-?	?	?	-∞
+0	+num	-num	+∞	-∞	+0	0	+?	?	∞	?	0
-0	+num	-num	+∞	-∞	0	-0	?	-?	∞	?	0
+?	+?	?	+?	?	+?	?	+?	?	?	?	?
-?	?	-num	?	-?	?	-?	?	-?	?	?	?
∞	∞	∞	?	?	∞	∞	?	?	∞	?	∞
?	?	?	?	?	?	?	?	?	?	?	?
0	+num	-num	+∞	-∞	0	0	?	?	∞	?	0

表-2. 減算の定義

op1 \ op2	+num	−num	+∞	−∞	+0	−0	+?	−?	∞	?	0
+num	±num ±0.0	+num +∞	−∞	+∞	+num	+num	?	+?	∞	?	+num
−num	−num −∞	±num ±0.0	−∞	+∞	−num	−num	−?	?	∞	?	−num
+∞	+∞	+∞	?	+∞	+∞	+∞	?	+?	?	?	+∞
−∞	−∞	−∞	−∞	?	−∞	−∞	−?	?	?	?	−∞
+0	−num	+num	−∞	+∞	0	+0	?	+?	∞	?	0
−0	−num	+num	−∞	+∞	−0	0	−?	?	∞	?	0
+?	?	+?	?	+?	?	+?	?	+?	?	?	?
−?	−?	?	−?	?	−?	?	−?	?	?	?	?
∞	∞	∞	?	?	∞	∞	?	?	?	?	∞
?	?	?	?	?	?	?	?	?	?	?	?
0	−num	+num	−∞	+∞	0	0	?	?	∞	?	0

表-3. 乗算の定義

op1 \ op2	+num	−num	+∞	−∞	+0	−0	+?	−?	∞	?	0
+num	+num +∞,+0	−num −∞,−0	+∞	−∞	+0	−0	+?	−?	∞	?	0
−num	−num −∞,−0	+num +∞,+0	−∞	+∞	−0	+0	−?	+?	∞	?	0
+∞	+∞	−∞	+∞	−∞	+?	−?	+?	−?	∞	?	?
−∞	−∞	+∞	−∞	+∞	−?	+?	−?	+?	∞	?	?
+0	+0	−0	+?	−?	+0	−0	+?	−?	?	?	0
−0	−0	+0	−?	+?	−0	+0	−?	+?	?	?	0
+?	+?	−?	+?	−?	+?	−?	+?	−?	?	?	?
−?	−?	+?	−?	+?	−?	+?	−?	+?	?	?	?
∞	∞	∞	∞	∞	?	?	?	?	∞	?	?
?	?	?	?	?	?	?	?	?	?	?	?
0	0	0	?	?	0	0	?	?	?	?	0

表-4. 除算の定義

op1 \ op2	+num	−num	+∞	−∞	+0	−0	+?	−?	∞	?	0
+num	+num +∞,+0	−num −∞,−0	+0	−0	+∞	−∞	+?	−?	0	?	?
−num	−num −∞,−0	+num +∞,+0	−0	+0	−∞	+∞	−?	+?	0	?	?
+∞	+∞	−∞	+?	−?	+∞	−∞	+?	−?	?	?	?
−∞	−∞	+∞	−?	+?	−∞	+∞	−?	+?	?	?	?
+0	+0	−0	+0	−0	+?	−?	+?	−?	0	?	?
−0	−0	+0	−0	+0	−?	+?	−?	+?	0	?	?
+?	+?	−?	+?	−?	+?	−?	+?	−?	?	?	?
−?	−?	+?	−?	+?	−?	+?	−?	+?	?	?	?
∞	∞	∞	?	?	∞	∞	?	?	?	?	?
?	?	?	?	?	?	?	?	?	?	?	?
0	0	0	0	0	?	?	?	?	0	?	?

5. 表現誤差

ある実数 x に対応する代表点 x^* は，仮数部の最後の桁の次の桁を丸めた数であるとされている．（我々のレベル0の表示も同様にする．レベル1以上の場合にも，仮数部および指数部に対して同様な考え方を適用できる．）このとき，"相対誤差"

$$e_{rep}(x) = \frac{|x^* - x|}{|x|} \qquad (5.1)$$

を x の"表現誤差"と呼ぶ．また，

$$u(x) = \max_y \{ e_{rep}(y) \mid y \text{は} x \text{と同じ代表点} x^* \text{を持つ} \} \qquad (5.2)$$

を"丸めの単位"と呼ぶ．β 進法で仮数部が ℓ 桁のとき，

$$u(x) = \frac{1}{2}\beta^{1-\ell} \qquad (5.3)$$

である．（丸めは四捨五入．）なお，非数 x については，$e_{rep}(x)$ は定義されない（あるいは ∞ である）とするのが"よい"であろう．（この意味でも，0や±0は非数とみなすのが適当である．）

異なる表現方式の特長を表現誤差，丸めの単位を用いて比較すると興味深い．一語長を64ビットと仮定し比較対象として以下の典型的なものを考える．

<1> IBM型：— 16進，仮数部56ビット（丸めは"切捨て"），指数部7ビット（"げたばき"式），符号1ビット．（"倍精度"実数型．）

<2> Kahan型：— 2進，仮数部52ビット（先頭ビットを隠すので実質的には53ビット；丸めは"0捨1入"），指数部11ビット（"げたばき"式），符号1ビット（"倍精度数"）．（正規化するとunderflowするとき，それ

を遅らせるための"非正規化数"も考える.) [8]

<3> Morris型:— Morrisの原論文[12]の精度を一語長が64ビットの場合に改作したもの. 2進,"指数部長"を表わす部分3ビット(これが表わす数をG(=0,1,…7)とする), 残り61ビットを仮数部, 指数部に割当てる. (丸めは0捨1入). 指数部(符号付き絶対値表示)の長さは,

(i) G+1 (Morris (i)型)

(ii) G+4 (Morris (ii)型)

の2通りの解釈が提案されている.

<4> 本論文の方式:— §3.2参照. レベル0を主として考える. (丸めは0捨1入).

正規化条件としては, <1>と<3>は(2,2)を, <2>と<4>とは(2,3)を用いる.

$|\log|x||$ が比較的小さい範囲での $e_{rep}(x)$ を図-3に, 広い範囲にわたっての $U(x)$ を図-4に示す.

これらの図からもわかるように, 本論文の方式は,"あふれ"が起こる"不連続点"がなく,"普通の大きさ"の数に対してはIBM型, Kahan型より誤差が小さく, 数が極端に大きくあるいは小さくなっても連続的に誤差が増加する形で対処している. このような点からみて, 本方式は各種の表現方式を比較検討する際の一つの基準となりうるであろう.

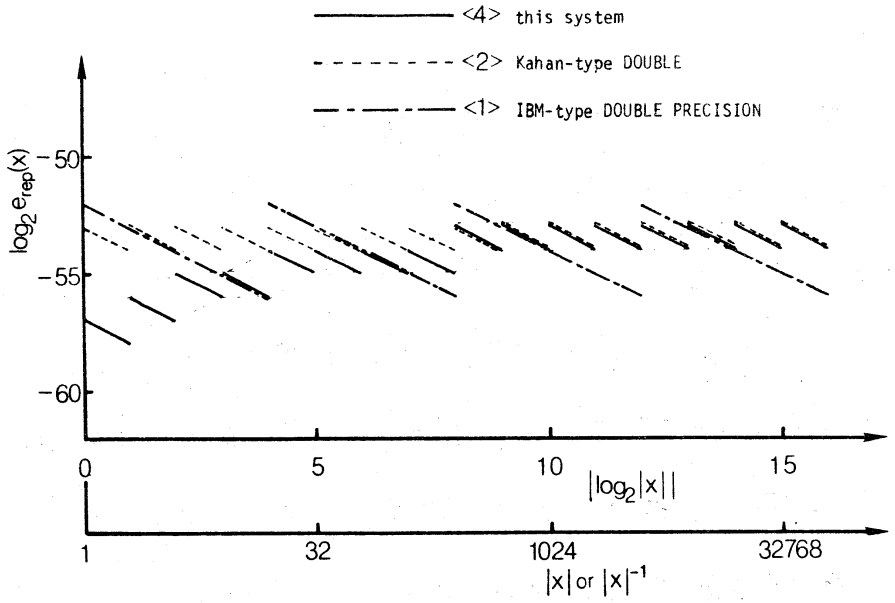


図3、各種表現方式による表現誤差 $e_{rep}(x)$ の比較

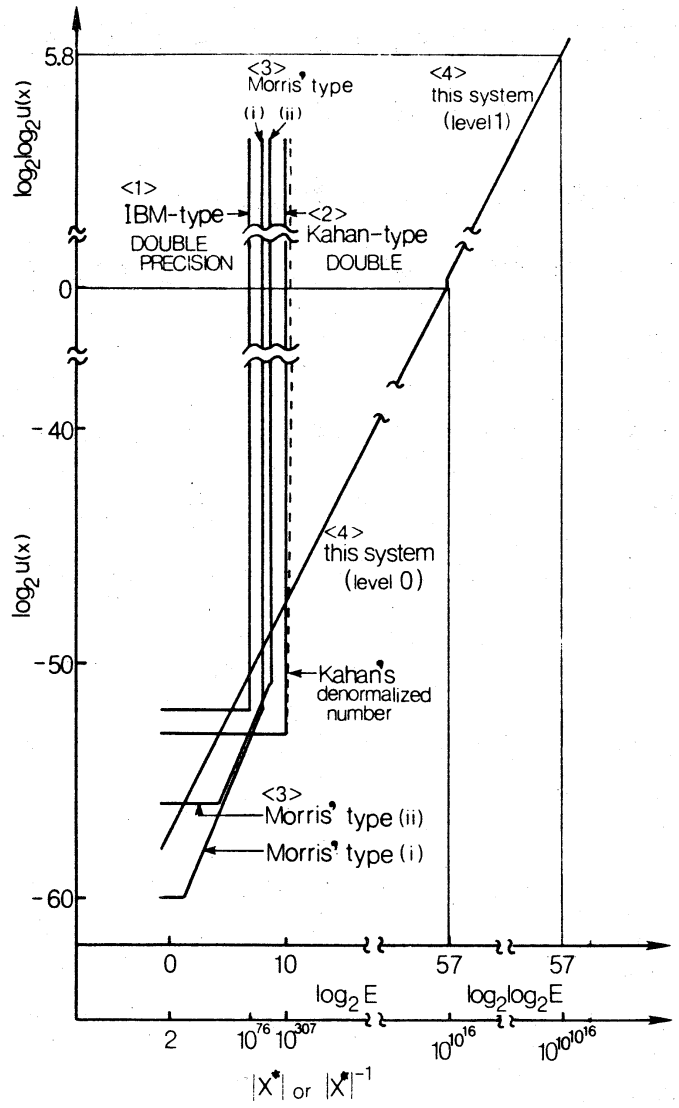


図-4、各種表現方式による内部単位 $u(x)$ の比較

6. 新体系による数値計算例

6.1. 実験プログラム

本論文で提案した方式を§3で述べた形で実現するプログラム(サブプログラム群)を東京大学の大型計算機センターのHITAC-8800/8700システムのFORTRANおよび教育用計算機センターのMelcom-Cosmo 900のFORTRANで作成した。(現在下記のものがある)、

1° 四則演算用サブルーチン(計約300行),

2° 入出力用関数副プログラム(計約200行),

3° 型変換(FORTRANのデータと本体系の間の変換を用う)用関数副プログラム(計約200行),

4° 平方根用関数副プログラム(計約60行),

5° 補助サブルーチン群(比較, 丸め, 仮数部と指数部の分離と結合等の処理)(計約300行)。

なお現在の版では非数は"0"のみが特殊なビットパターンとして含まれている。

プログラムは実験用であり, 速度向上の為の技巧を凝らしていないので, 速度は遅いが(通常の倍精度実数型演算と比較して約600~1000倍程度の時間を要する(HITACでの比較)), この点は大幅な改良の余地がある。特に演算回路を金物化した場合には, IBM型やKahan型の1割増し程度の時間で実現できる見込みがある。

6.2. 数値計算例

極端に大きな数，小さな数が出現する典型的な数値計算の例として代数方程式のGraeffe法[13]による解を考えてみる．簡単のため，正の零点のみの多項式方程式

$$P_1(x) = x^8 - 11x^7 + 45.35x^6 - 88.55x^5 + 86.752x^4 - 43.274x^3 + 10.981x^2 - 1.32x + 0.0576$$
$$= (x-0.1)(x-0.2)(x-0.3)(x-0.4)(x-1)(x-2)(x-3)(x-4) \quad (6.1)$$

および

$$P_2(x) = x^4 - 10.438575930206 13x^3 + 40.587405675 8x^2 - 69.6040857054 5396x + 44.3671561490 6059$$
$$= (x-2)(x-e)(x-\sqrt{7.4})(x-3) \quad (6.2)$$
$$(e = 2.7182818 28\cdots,\ \sqrt{7.4} = 2.7202941017\cdots)$$

を考える．

$P(x) = x^n - a_{n-1}x^{n-1} + \cdots + (-1)^{n-1}a_1 x + (-1)^n a_0$ の零点を $\alpha_i\ (i=1,\cdots,n)$ とするとき，$\alpha_i^{2^\nu}\ (i=1,\cdots,n)$ を零点とする多項式

$$P^{(\nu)}(x) = x^n - a_{n-1}^{(\nu)}x^{n-1} + \cdots + (-1)^n a_0^{(\nu)} \quad (6.3)$$

は簡単な漸化式

$$a_k^{(0)} = a_k,\quad a_k^{(\nu+1)} = \sum_{i+j=2k} (-1)^i a_i^{(\nu)} a_j^{(\nu)},\quad (\nu = 0,1,\cdots) \quad (6.4)$$

で計算され，α_i の近似値は

$$\alpha_i \fallingdotseq \left(a_{i-1}^{(\nu)} / a_i^{(\nu)}\right)^{1/2^\nu} \quad (6.5)$$

で計算できる．

$P_1(x)$に対する反復計算の結果を図-5に，$P_2(x)$に対するものを図-6に示す．また それらをまとめたものを図-7に示す．

図-5においては$n=7$くらいですべての零点が十分な精度で得られており，(その後無駄な反復を行なっても精度が悪化しないことに注目すべきである．) IBM型のものでは$n=6$くらいで破綻をきたす．

図-6においては，近接零点があるため，必要な反復回数が増加し，また最終的に得られる零点の近似値の精度もいくらか劣る．しかし，本方式では$n=16$くらいで，10進10桁の精度で零点が得られている．IBM型のものでは，$n=6$くらいで破綻をきたし(そのときの近接零点の精度は最悪のものでは僅か2桁)，Kahan型のものでも$n=7$までしかもたない(そのときの近接零点の精度は，最悪のものでは2桁)．

7. 今後の課題

本論文では，新しい浮動小数点方式の基本的な着想を提案し，その特長を従来の諸方式と比較考察し，それを実現する試作プログラムによる数値計算例をあげた．しかしこの方式については更に検討すべき課題が数多く残されているので，それらについて逐次検討結果を発表してゆきたい．主なものは次の通りである．

(1) 演算の全物化：一本来この種の方式は演算を全物化することによって始めてその特長が発揮できる．そのための演算回路の設計試作を行うこと．理論的には従来方式とほぼ同程度の速度が得

Kahan型の限界

ν = 1

3.317760000000000 EE-3	=A(1)
- 4.770432000000000 EE-1	=A(2)
1.639877247999999 EE 1	=A(3)
- 1.954100328000000 EE 2	=A(4)
8.294775057599999 EE 2	=A(5)
- 9.027198200000001 EE 2	=A(6)
2.820272999999999 EE 2	=A(7)
- 3.030000000000000 EE 1	=A(8)
8.339569428721113 EE-2	=ROOT(1)
1.705584280072372 EE-1	=ROOT(2)
2.896891579541276 EE-1	=ROOT(3)
4.853679799876546 EE-1	=ROOT(4)
9.585743805733726 EE-1	=ROOT(5)
1.789084751052074 9 EE0	=ROOT(6)
3.050873921218035 8 EE0	=ROOT(7)
5.504543577809154 0 EE0	=ROOT(8)

ν = 7

2.158131372982 4 EE-159	=A(1)
- 2.158131372982 48 EE-31	=A(2)
6.342178093180 84 EE 58	=A(3)
- 5.379201700675 8 EE 125	=A(4)
4.645569257886 8 EE 176	=A(5)
- 4.645569257887 0 EE 176	=A(6)
1.365210104749 8 EE 138	=A(7)
- 1.157920892373 27 EE 77	=A(8)
9.999999999999997 EE-2	=ROOT(1)
2.000000000000006 EE-1	=ROOT(2)
2.999999999999984 EE-1	=ROOT(3)
4.000000000000013 EE-1	=ROOT(4)
9.999999999999996 EE-1	=ROOT(5)
2.000000000000022 EE0	=ROOT(6)
2.999999999999951 EE0	=ROOT(7)
4.000000000000027 EE0	=ROOT(8)

収束

ν = 5

2.15535826671273 EE-40	=A(1)
- 2.155358267214587 EE-8	=A(2)
5.018346158 19841 EE 14	=A(3)
- 2.708464085 10473 EE 31	=A(4)
1.468113846 64593 EE 44	=A(5)
- 1.468113846 98750 EE 44	=A(6)
3.418226842 32871 EE 34	=A(7)
- 1.844859709 81938 EE 19	=A(8)
9.999999999927237 EE-2	=ROOT(1)
1.999999855136427 EE-1	=ROOT(2)
2.999990800367525 EE-1	=ROOT(3)
4.000012555971386 EE-1	=ROOT(4)
9.999999999927293 EE-1	=ROOT(5)
1.999999855136424 0 EE0	=ROOT(6)
2.999990800367536 2 EE0	=ROOT(7)
4.000012555971390 5 EE0	=ROOT(8)

ν = 8

4.65753102305 09 EE-318	=A(1)
- 4.65753102305125 EE-62	=A(2)
4.022322296562 3 EE 117	=A(3)
- 2.893581093655 3 EE 251	=A(4)
2.158131372982 3 EE 353	=A(5)
- 2.158131372982 5 EE 353	=A(6)
1.863798630110 9 EE 276	=A(7)
- 1.340780792994 5 EE 154	=A(8)
9.999999999999997 EE-2	=ROOT(1)
2.000000000000006 EE-1	=ROOT(2)
2.999999999999984 EE-1	=ROOT(3)
4.000000000000013 EE-1	=ROOT(4)
9.999999999999996 EE-1	=ROOT(5)
2.000000000000022 EE0	=ROOT(6)
2.999999999999951 EE0	=ROOT(7)
4.000000000000027 EE0	=ROOT(8)

IBM計算機

ν = 6

4.64556925788690 EE-80	=A(1)
- 4.64556925788700 EE-16	=A(2)
2.518368140929 56 EE 29	=A(3)
- 7.334304199605 79 EE 62	=A(4)
2.155358266712 71 EE 88	=A(5)
- 2.155358266712 76 EE 88	=A(6)
1.168422057633 48 EE 69	=A(7)
- 3.402823703546 38 EE 38	=A(8)
9.999999999999997 EE-2	=ROOT(1)
1.999999999999838 EE-1	=ROOT(2)
2.999999999527234 EE-1	=ROOT(3)
4.000000000630681 EE-1	=ROOT(4)
9.999999999999996 EE-1	=ROOT(5)
1.999999999999 8341 EE0	=ROOT(6)
2.999999999527 2459 EE0	=ROOT(7)
4.000000000630 6709 EE0	=ROOT(8)

ν = 30

2.298526 EE-1330986224	=A(1)
- 2.2985271 EE-257244400	=A(2)
5.4763732 EE 493268927	=A(3)
- 2.670771 EE 1054705705	=A(4)
1.516089 EE 1481990536	=A(5)
- 1.516090 EE 1481990536	=A(6)
3.612177 EE 1158762039	=A(7)
- 1.7616144 EE 646456993	=A(8)
9.999999999999999 EE-2	=ROOT(1)
2.000000000000006 EE-1	=ROOT(2)
2.999999999999984 EE-1	=ROOT(3)
4.000000000000013 EE-1	=ROOT(4)
9.999999999999994 EE-1	=ROOT(5)
2.000000000000024 EE0	=ROOT(6)
2.999999999999953 EE0	=ROOT(7)
4.000000000000027 EE0	=ROOT(8)

図-5. $P_1(x)$の零点を求めるGraeffe反復

Kahan型の限界

ν = 1

1.9894445447551 23 EE 3	=A(1)
- 1.2432332162788632 EE 3	=A(2)
2.8293674441918649 EE 2	=A(3)
- 2.7789056090893071 EE 1	=A(4)
1.2583031670239898 EE0	=ROOT(1)
2.0961947379335457 EE0	=ROOT(2)
3.1908602092232556 EE0	=ROOT(3)
5.2715326138544105 EE0	=ROOT(4)

ν = 7

6.6662837510202 EE 210	=A(1)
- 1.9500447225817 EE 172	=A(2)
9.6227697791204 EE 116	=A(3)
- 1.1790266194857O EE 61	=A(4)
1.9999999999999513 EE0	=ROOT(1)
2.7045783307732R1 EE0	=ROOT(2)
2.7340770775614731 EE0	=ROOT(3)
3.0000001622446730 EE0	=ROOT(4)

ν = 5

5.0812545154R211 EE 52	=A(1)
- 1.1832016953495T EE 43	=A(2)
3.0253933010615T EE 29	=A(3)
- 2.0128425620976 3 EE 15	=A(4)
1.9999931360575824 EE0	=ROOT(1)
2.6592468559773725 EE0	=ROOT(2)
2.7735140870227926 EE0	=ROOT(3)
3.0077660744101009 EE0	=ROOT(4)

ν = 16

6.7375701284 EE 107941	=A(1)
- 3.36284975669 EE 88213	=A(2)
4.01365489311 EE 59751	=A(3)
- 4.15479228694 EE 31268	=A(4)
1.9999999999999518 EE0	=ROOT(1)
2.7182818285951764 EE0	=ROOT(2)
2.7202941016100686 EE0	=ROOT(3)
3.0000000000009399 EE0	=ROOT(4)

IBM型の限界 ↑ 収束

ν = 6

2.5819147451107 EE 105	=A(1)
- 1.3996587960106 5 EE 86	=A(2)
4.3898173247259 3 EE 58	=A(3)
- 3.4464565195794 2 EE 30	=A(4)
1.9999999998191536 EE0	=ROOT(1)
2.6899448172248597 EE0	=ROOT(2)
2.7487913512240691 EE0	=ROOT(3)
3.0001740482911321 EE0	=ROOT(4)

ν = 30

1.577772 EE 1768518918	=A(1)
- 3.759240 EE 1445290421	=A(2)
2.5108124 EE 978970272	=A(3)
- 2.0511859 EE 512305046	=A(4)
1.9999999999999518 EE0	=ROOT(1)
2.7182818285951764 EE0	=ROOT(2)
2.7202941016100686 EE0	=ROOT(3)
3.0000000000009399 EE0	=ROOT(4)

図-6、 $P_2(x)$の零点を求めるGraeffe反復

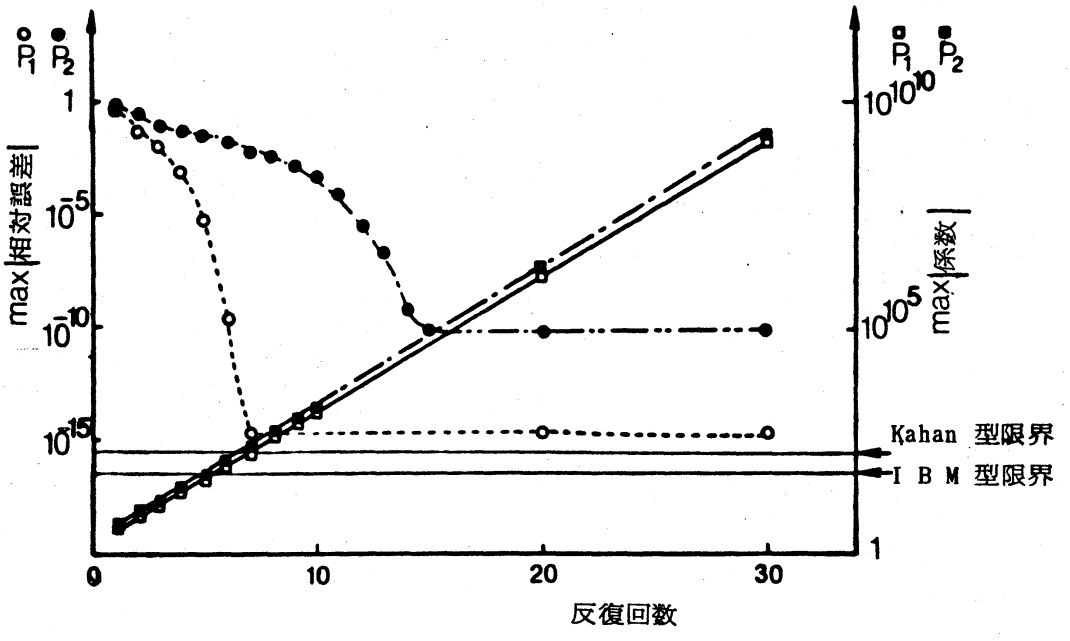


図-7、 Graeffe法による反復過程

られる見込みがある．

(ii) 新表現方式に適した丸め誤差理論の確立：一本論文では数の表現に直接関連した誤差のみを論じたが，「表現誤差がほぼ一定」という従来の誤差理論[13]の仮定の成立しない本方式における，四則その他の演算に伴う誤差の拡大，累積に関する理論を確立すること．従来の理論が"非数"の場合を含めてどのように変更されるかを調べることは重要であろう．

(iii) 高次レベル，非数の実際的な取扱い：一非数やレベル1以上の入出力をどうするか，どのようにして実現するか，またどのくらいのレベルまで考えるべきか，等々について，理論，実際面から検討の余地がある．

8. おわりに

本研究に関しては，非公式研究会などにおいて，電気通信大学教授森口繁一先生，慶応大学教授相磯秀夫先生はか多くの方々から貴重な御助言を賜り，また研究室の方々からも有用な討論を頂いた．それらにより内容の多くの部分を改良することができた．ここに記して，感謝の意を表わす次第である．

なお，本研究は文部省科学研究費の援助による部分を含んでいる．

9. 参考文献

[1] Brent, R. P., On the Precision Attainable with Various Floating-Point Number System. IEEE Trans. on Computers, Vol. C-22, No. 6, pp. 601-607(1973).

[2] Cody, W. J., Jr., Static and Dynamic Numerical Characteristics of Floating-Point Arithmetic. IEEE Trans. on Computers, Vol. C-22, NO. 6, pp. 598-601(1973).

[3] Edger, A. D., and Lee, S. C., FOCUS:Microcomputer Number System. Comm. ACM, Vol. 22, No. 3, pp. 166-177(1979).

[4] Hamming, R. W., On the Distribution of Numbers. The Bell System Technical Journal, Vol. 40, No. 8, pp. 1609-1625(1970).

[5] Hehner, E. C. R., and Horspool, R. N. S., A New Representation of the Rational Arithmetic. SIAM J. on Comput., Vol. 8, No. 2, pp. 124-134(1979).

[6] 一松信, 新標準浮動小数点体系の提案. 情報処理, Vol. 20, No. 9, pp. 793-797(1979).

[7] Ida, T., and Goto, E., Overflow Free and Variable Precision Computing in FLATS. Journal of Information Processing, Vol. 1, No. 3, pp. 140-142(1978).

[8] Kahan, W., and Palmer, J., On a Proposed Floating-Point Standard, ACM SIGNUM Newsletter, Special Issue, pp. 13-21(Oct. 1979).

[9] Knuth, D. E., The Art of Computer Programming, Vol. 2:Seminumerical Algorithms. Addison-Wesley, Reading, Massachusetts(1969).

[10] Morris, R., Tapered Floating Point: A New Floating Point Representation. IEEE Trans. on Computers, Vol. C-20, No. 6, pp. 1678-1679(1973).

[11] Reinsh, C. H., Principle and Preference for Computer Arithmetic. ACM SIGNUM Newsletter, Vol. 14, No. 1, pp. 12-27(1979).

[12] Swartzlander, E. E., Jr., and Alexopoulos, A. G., The Sign/Logarithm Number System. IEEE Trans. on Computers, Vol. C-24, No. 12, pp. 1238-1242(1975).

[13] Wilkinson, J. H., Rounding Errors in Algebraic Processes. Prentice-Hall, Englewood-Cliffs (1963).